Failure-Inference-Based Fast Reroute with Progressive Link Metric Increments

Phani Krishna Penumarthi, Aaron Pecora, Jason M. O'Kane, Srihari Nelakuditi
Computer Science and Engineering, University of South Carolina
{penumarp, apecora, jokane, srihari}@sc.edu

Abstract—**This paper is focused on providing fast reroute and loop-free convergence in traditional IP networks, without making any modifications to the IP datagram and without requiring any coordination between routers for FIB updates. Failure inference based fast route (FIFR) is an approach in which routers adjacent to a failed link or router perform local rerouting around the failure, without notifying non-adjacent routers about the failure. The non-adjacent routers utilize interface-specific forwarding tables, which are precomputed based on potential inferred failures that could cause a packet for a given destination to arrive through that unusual interface, to ensure loop-free forwarding towards the destination. However, as long as the failure lasts, packets that were to be forwarded over the failed link traverse suboptimal paths, as they reach the router adjacent to the failure and then are rerouted along a detour. Therefore, in case of a long-lasting failure, it is desirable to trigger a network-wide link state update, so that all routers can converge to new optimal forwarding tables. But, without some coordination between routers to install their forwarding entries in a specific order, there may be transient forwarding loops during the convergence period. As we are interested in a mechanism that does not require any such coordination between routers, we consider the possibility of employing progressive link state updates. In this paper, we show that FIFR with progressive link metric increments can guarantee loop-free forwarding not only before/after but during convergence too and protect against non-partitioning single link failures.**

I. Introduction

The key objectives of any intra-domain routing protocol such as OSPF are reachability, i.e., forward packets to their destinations, and optimality, i.e., along the shortest paths. To ensure optimality, any changes in the link state need to be propagated across the network, so that all routers recompute new shortest paths and install corresponding forwarding entries. On the other hand, to respond to failures quickly, due to relatively long convergence delay associated with a link state update, optimality is often traded-off for reachability, by having routers adjacent to failures perform local rerouting, without invoking the control plane. However, when a failure lasts beyond a certain duration, it is desirable to trigger a link state update so that all routers recompute the optimal routes in the new topology. But a straightforward link state update can cause transient forwarding loops during the convergence process, unless routers coordinate to install their forwarding entries in a particular order. In this paper, we study the feasibility of developing a combination of local reroute and global update mechanisms that can achieve loop-free convergence, while performing disruption-free forwarding around a failed link, without requiring any modification to the IP datagram and without needing any additional coordination between routers.

There have been numerous proposals for performing fast reroute in traditional IP networks [1]–[5]. Many of these schemes require encapsulation [1] or modification to the IP header to carry additional bits of information [2]. Among all the IP fast reroute schemes that provide full protection against any single non-partitioning failure, we consider *failure inference based fast route* (FIFR) [6], as it is an approach that does not need any changes to the IP datagram. Under FIFR, routers adjacent to a failed link or router perform local rerouting around the failure, without notifying non-adjacent routers about the failure. The non-adjacent routers utilize *backwarding* entries, which are associated with each interface along the reverse shortest path and are precomputed based on potential inferred failures that could cause a packet for a given destination to arrive at that interface, to ensure loop-free forwarding to the destination. While FIFR is quite suitable for short-lived failures, forwarding packets to the point of failure and then rerouting them is undesirable for longer-lasting failures. In such cases, it is preferable to trigger a link state update and initiate re-convergence to optimal routes.

During the convergence period, i.e., between the time a link state update is initiated and the time all routers install new forwarding entries, packets may be forwarded by some routers based on the new topology and others based on the old topology, potentially leading to forwarding loops. To prevent loops, [7] proposed a scheme that imposes a certain order between the FIB updates of different routers. While it guarantees loop-free convergence, it requires some form of coordination among routers to order their updates. As we are interested in an approach that does not require coordination between routers, we consider the progressive link metric increment method [8], which sends a sequence of updates such that each update is loop-free. The question this paper explores is whether progressive updates can be used in conjunction with FIFR to achieve loop-free convergence while performing fast reroute.

We observe that FIFR, due to backwarding entries, mitigates the problem of forwarding loops during convergence with traditional unordered updates. However, it does not completely eliminate the looping problem. Even with progressive link metric increments, we find that simultaneous updating of both interface-independent forwarding entries and interface-specific backwarding entries at a router can cause forwarding loops in some instances. However, by decoupling the updating of forwarding and backwarding entries, it is possible to ensure continuous loop-free forwarding while converging to optimal routing. We refer to this approach as FIFR++.

The rest of the paper is organized as follows. Section II presents the background material on both the FIFR approach and progressive link state updates. Section III describes the proposed FIFR++ approach of employing progressive link metric increments in conjunction with FIFR and proves that FIFR++ is loop-free while protecting against single link failures. The limitations of this work are discussed in Section IV. Section V reviews the related work that motivated the design of FIFR++. Finally, we conclude the paper in Section VI.

II. Background

In this section, we present some background material about the schemes upon which the proposed approach FIFR++ is built, and then describe FIFR++ in the following section.

A. Convergence with Progressive Link State Updates

An obvious approach to deal with any link state changes is to propagate them network-wide so that all routers can recompute their next hops and forward packets along the optimal routes. The problem is that, during the convergence period, i.e., between the time a link state update is initiated and the time all routers install new forwarding entries, packets may be forwarded by some routers based on new state and others based on old state, potentially causing forwarding loops.

Consider the topology in Fig. 1, where each link is labelled with its cost. Suppose the link B−E is being shut down for maintenance by changing its cost from 1 to ∞, and all routers are notified of this change. Due to the delays in the propagation of the link state advertisements and recomputation/installation of new forwarding entries, it is possible that B is in the *ac* (after change) era, i.e., starts forwarding according to the new cost, whereas A is still in the *bc* (before change) era, i.e., continues forwarding based on the old cost. Then, as per the new shortest path B→A→C→E, a packet destined for E from B is forwarded to A. Being in the *bc* era, router A, as per the old cost, forwards it back to B, resulting in a loop.

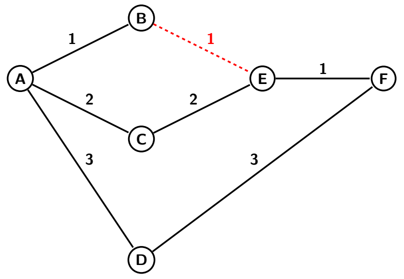

Fig. 1. A topology used for illustration of progressive updates and FIFR.

To prevent such forwarding loops during convergence, [7] proposed a scheme that imposes a certain order among the FIB updates of different routers. In this example, B installs new FIB entries only after A has done the same (and the link B−E is shut down only after its adjacent nodes B and E update their FIBs). Then, a packet from B is forwarded to E, either along the old path B→E before the update or the new path B→A→C→E after the update, avoiding loops at any time instant. Enforcing such an order requires some form of coordination between routers. As we are interested in an approach that does not require any such coordination, we consider the progressive link metric increments approach proposed in [8].

The main idea behind the progressive link metric increments is as follows. Instead of sending an update with B−E cost of ∞, suppose we send a sequence of updates with a cost of 2, 3, and so on, each with a progressively higher cost, until B−E is not along any shortest path. Note that a subsequent update is sent only after the network has converged to the previous update. Consider the first update in this sequence, where the cost of B−E is increased from 1 to 2. With the old cost of 1 or the new cost of 2, the shortest paths from A to E and B to E remain the same, and hence this update would not cause a forwarding loop, even if A and B are in different eras.

When B−E cost is updated to 3, A→C→E becomes a shortest path with the same cost as that of A→B→E, while the shortest path from B to E remains B→E. Again, regardless of the eras A and B are in, this update does not cause a forwarding loop. Now, by increasing B−E cost to 4, A→B→E ceases to be a shortest path. The following update of cost to 5 makes A one of the next hops from B to E. Since B is no longer a next hop from A to E after the previous update, this update does not cause any back and forth between A and B. Next, updating the B−E cost to 6 or higher, effectively eliminates the link from the topology, since it is not along the shortest path between its adjacent nodes. We can then send a final update with B−E cost of ∞, to inform all routers that the link is actually down. Thus, each update yields loop-free convergence and progressively brings the network to the desired target topology without B−E, allowing the B−E link to be shut down gracefully.

It is not necessary that the link cost has to be strictly incremented in steps of 1 to avoid loops. It has been shown that loop-freedom can be assured with a shorter metric sequence [8], which is specific to each link and depends on the topology. Table I shows the sequence of metric updates needed to gracefully bring down a link in the topology in Fig. 1.

B. Failure Inference based Fast Reroute

We now explain the core idea behind the FIFR approach using a simple example. We refer the reader to [3], [6] for the full details. Suppose the link between routers B and E is down, and only B and E are aware of the failure. Imagine forwarding a packet from source A to destination F. Based on the link costs, the shortest path from A to F is A→B→E→F. So, router A

TABLE I
METRIC SEQUENCE REQUIRED TO BRING EACH LINK DOWN, FOR THE TOPOLOGY IN FIG. 1

Link	Metric Sequence
A–B	2, 3, 4, ∞
A–C	3, 4, ∞
A–D	4, 5, 6, ∞
B–E	2, 3, 4, 5, ∞
C–E	3, 4, ∞
D–F	4, 5, 6, ∞
F–E	2, 3, 4, 5, 6, 7, 8, ∞

TABLE II
KEY LINKS AND BACK HOPS FOR UNUSUAL INCOMING INTERFACES FOR THE TOPOLOGY IN FIG. 1

Interface	Destination	Key Link	Back Hop
B→A	E	B–E	C
B→A	F	E–F	D
A→B	C	A–C	E
A→B	D	A–D	E
E→B	C	C–E	A
E→B	F	E–F	A
A→C	B	A–B	E
E→C	B	B–E	A
F→D	E	E–F	A
B→E	A	A–B	C
F→E	D	D–F	B

will forward the packet to its next hop B. Router B, being adjacent to the failed link B–E, initiates local rerouting. Since shortest path to F, without the B–E link, is B→A→C→E→F, it forwards the packet back to A. Normally, this would cause the packet destined for F to go back and forth between routers A and B, resulting in a forwarding loop.

Under FIFR, however, A infers potential failures along the shortest path to F that would cause the packet to arrive at A from its usual next hop B (i.e., through the *unusual* incoming interface B→A). It is apparent that the failure of link B–E would cause the packet for F to arrive through interface B→A. Similarly, the failure of E–F would also cause the packet destined for F to arrive through interface B→A, as the shortest path from E to F without link E–F is E→B→A→D→F. Since A does not know which of these links actually failed, it excludes all such candidate links to find an alternate next hop (which we refer to as *back hop*), which is D, for packets destined for F arriving through interface B→A.

We observe that the resulting back hop would be the same, if we were to exclude just one of those candidates links, referred to as the *key link*, that is closest to the destination. In this example, the key link for destination F and interface B→A is E–F. Router A can precompute a key link per destination for each of its interfaces (which may be none if there are no candidate links), and compute the corresponding back hops. Table II lists the key links and back hops for all combinations of unusual incoming interfaces and destinations in Fig. 1.

Effectively, FIFR computes interface-specific forwarding entries, i.e., a packet's next hop depends not only on its destination but also on the incoming interface. Most of these entries would be identical to usual forwarding entries based on destination only, independent of the incoming interface. Only the unusual interfaces that lie along the reverse shortest path from the point of failure to the destination have back hops.

To put this formally, under FIFR, each router i has a forwarding entry $\mathcal{F}_i^d$ per each destination d. In addition, it keeps a *backwarding* entry $\mathcal{B}_{j\to i}^d$ per each destination d and neighbor $j \in \mathcal{F}_i^d$ (both $\mathcal{F}_i^d$ and $\mathcal{B}_{j\to i}^d$ are sets, as there could be multiple shortest paths of equal cost). Let $\mathcal{K}_{j\to i}^d$ be the farthest link along a shortest path from i to d, whose failure would cause the packet to d arrive through interface $j\to i$. While $\mathcal{F}_i^d$ is the set of usual next hops from i to d, $\mathcal{B}_{j\to i}^d$ is the set of next hops from i to d without the link $\mathcal{K}_{j\to i}^d$. When $\mathcal{K}_{j\to i}^d$ is $\varnothing$, $\mathcal{B}_{j\to i}^d$ is the same as $\mathcal{F}_i^d$. A packet originating at i to destination d is forwarded to $\mathcal{F}_i^d$. A packet destined for d arriving at i through neighbor j is forwarded to $\mathcal{B}_{j\to i}^d$ if $j \in \mathcal{F}_i^d$, otherwise to $\mathcal{F}_i^d$.

The forwarding and backwarding entries can effectively combined into one interface-specific forwarding table per interface, as shown in Table III for router A. Note that most of the entries are usual next hops along the shortest paths to the destinations. Only for the interface B→A, the next hops for destinations E and F are different from the usual forwarding entries. It is also important to emphasize that these entries *are computed* a priori *and not on the fly*. Moreover, since routers nowadays maintain a copy of the FIB at each interface to perform forwarding at line speed, interface-specific forwarding can be implemented without any changes to the forwarding plane.

TABLE III
INTERFACE-SPECIFIC FORWARDING ENTRIES AT ROUTER A FROM THE TOPOLOGY IN FIG. 1

Interface	Destination				
	B	C	D	E	F
B→A	B	C	D	$\textcircled{\textit{C}}$	$\textcircled{\textit{D}}$
C→A	B	C	D	B	B
D→A	B	C	D	B	B

It has been shown that FIFR, by employing interface-specific forwarding, can guarantee loop-free forwarding to all reachable destinations in case of a single link or router failure [3], [6]. However, because the packets are rerouted along alternate paths only from the point of failure, the resulting paths are suboptimal. For instance, in the above example, when link B–E is down, a packet from A to F traverses the path A→B→A→D→F, compared to the optimal path A→C→E→F. Therefore, it is desirable to initiate a link state update in case of a long-lasting failure, while performing fast reroute, so that all routers can converge to optimal routes.

III. Proposed Approach

We have already shown in Section II-A that, without FIFR, a straightforward updating of B−E's cost from 1 to ∞ causes transient forwarding loops during convergence. FIFR, however, helps mitigate this problem. Again, consider the scenario, where A is in the *bc* era and B is in the *ac* era, that leads to a loop without FIFR. But with FIFR, when B forwards a packet destined for E to A, instead of forwarding it back to B, using interface-specific forwarding, A will forward that to C. We find that in case of the topology in Fig. 1, updating about any other link failure, not just B−E, does not cause loops during convergence with FIFR, even with traditional link state updates, regardless of the order of routers updating their FIBs.

The above observation does not hold across all topologies. Consider the topology shown in Fig. 2, where the link I−M is down and a link state update is triggered with cost ∞. Assume that J, K, and L have recomputed their forwarding tables and are in the *ac* era, whereas the rest are in the *bc* era. Suppose J has a packet for M. As per the *ac* topology, the shortest path is J→L→N→M, and so J forwards it to L. L, which is also in the *ac* era, forwards it to N. According to N, which is in the *bc* era, the usual shortest path is N→L→J→H→I→M. Based on its inference on *bc* view, only the failure of J−L can cause a packet for M to arrive through L→N interface along the reverse shortest path. Therefore, as the backward path, without J→L, is N→O→K→I→M, the packet is forwarded to O, which in turn forwards to K. Since K is in the *ac* era, its shortest path to M, without I−M, is K→J→L→N→M. Hence, K forwards to J, which again forwards to L, causing a loop.

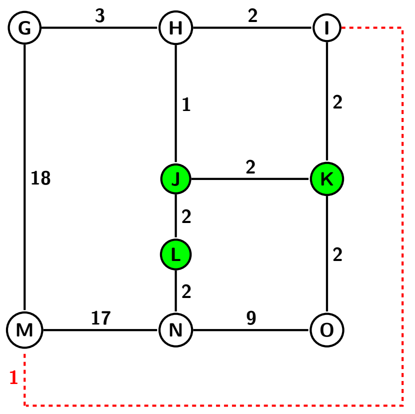

Fig. 2. An illustration of looping during convergence with FIFR when the link I−M is down. J, K, and L are in the *ac* era and the rest are in the *bc* era.

We propose to employ progressive link state updates along with FIFR to provide loop-free forwarding not only before/after convergence but during convergence as well. Note that while progressive updates alone can help gracefully shut down links for planned maintenance, FIFR facilitates instant shutdown of links and also protects against unplanned failures. The looping scenario discussed above does not occur with this approach, since the cost of I−M is increased gradually with a sequence of updates (the exact metric sequence is 10, 11, 13, 14, 18, 19, 20, 21, 22, 23, ∞). Imagine the first update, where the cost of I−M is set to 10. With this cost, regardless of the eras each router is in, packet from J to M is forwarded to I, which then reroutes it to H, along the path I→H→G→M. H infers the failure of I−M, as the packet to M arrives through unusual interface I→H, it forwards it to G. The router G does the same and forwards it to the destination M. Thus, with each update step, one or more source-destination pairs' shortest paths are made equal to that passing through I−M or turned away from I−M, while packets arriving at I are rerouted to the destination along a loop-free path with the help of FIFR.

There is a caveat, however, that a router, when it receives a link cost increase advertisement, should update its usual forwarding entries but not its backwarding entries for unusual incoming interfaces too. While it does not matter for the two topologies we discussed earlier, updating back hops can not guarantee loop-freedom in all instances. Consider the topology in Fig. 3, where the T−U link is down. Suppose the T−U cost is being progressively updated and currently the cost is being changed from 4 to 5. When S computes new tables, with cost 5, the shortest path to destination W would be S→R→T→U→W. Also, U−W ceases to be a key link and none of the other links along its shortest path would be a key link, since any of their failures would not cause the packet to arrive at S through the interface R→S. Therefore, for all interfaces, next hop from S to W would be set to R. On the other hand, R, which is still in the *bc* era, has T as its next hop for W. Also, S is the back hop to W for T→R interface (since U−W is a key link with T−U cost of 4 as its failure would cause the packet for W to arrive at R through T→R interface). In this scenario, a packet from S to W takes the path S→R→T and then gets rerouted along T→R→S. Because there is no key link associated with R→S, router S forwards it to R, forming a loop.

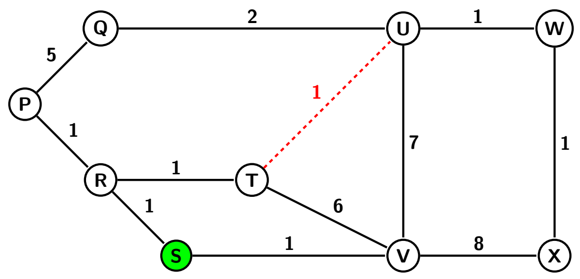

Fig. 3. A looping scenario with simultaneous updating of forwarding and backwarding entries during convergence with FIFR. Link T−U is down and it's cost is progressively updated from 4 to 5. Only router *S* is in the *ac* era.

We propose to eliminate the possibility of such loops while backwarding during convergence by defering the updating of back hops. In the above example, during progressive link metric increments, S changes its usual forwarding entries, but keeps the back hop for W in the table of R→S interface, which is along the reverse shortest path to W, to be V (since U−W is a key link with the original T−U cost of 1 as its failure would cause the packet for W to arrive at S through R→S interface). At the end of progressive updates, each router can

TABLE IV
NOTATION

$\mathcal{F}_i^d$	set of next hops from i to d
$\mathcal{B}_{j\to i}^d$	set of back hops from $j\to i$ to d
$\mathcal{K}_{j\to i}^d$	key link corresponding to $j\to i$ to d
C_{u-v}	cost of the link $u-v$
$\mathcal{T}_i$	shortest path tree (SPT) rooted at i
$\mathcal{P}(\mathcal{T}, d)$	shortest path from root of $\mathcal{T}$ to d
$\mathcal{D}(\mathcal{T}, d)$	shortest distance from root of $\mathcal{T}$ to d

independently, without any need for synchronization, install its new back hops corresponding to the topology without the failed link. We refer to this approach as FIFR++.

Operations under FIFR++ can be summarized as shown in Algorithm 1. The notation used is given in Table IV. A key aspect of FIFR++ is that while the forwarding entries at a router i, $\mathcal{F}_i$, are recomputed upon receiving each progressive link state update, its backwarding entries corresponding to each neighbor j, $\mathcal{B}_{j\to i}$ are recomputed only when the link is finally shut down.[1] The advantage of FIFR++ is that it guarantees loop-free forwarding during convergence too in case of single link failures. We prove this in the next section.

Algorithm 1 : Operations at router i under FIFR++

```
1:  if Router $i$ originates a packet $p$ to destination $d$
2:      Forward $p$ to $\mathcal{F}_i^d$
3:
4:  if Router $i$ receives a packet $p$ to $d$ from neighbor $j$
5:      if $j \in \mathcal{F}_i^d$
6:          Forward $p$ to $\mathcal{B}_{j\to i}^d$
7:      else
8:          Forward $p$ to $\mathcal{F}_i^d$
9:
10: if Router $i$ detects a failure of an adjacent link to $v$
11:     Recompute $\mathcal{F}_i$ without $i-v$
12:     Send LSAs to progressively update $C_{i-v}$ to $\infty$
13:     Recompute $\mathcal{B}_{j\to i}$ without $i-v$ for each neighbor $j$
14:
15: if Router $i$ receives an LSA with new cost $c$ for link $u-v$
16:     $C_{u-v} \leftarrow c$
17:     Recompute $\mathcal{F}_i$ with new cost of $u-v$
18:     if $c = \infty$
19:         Recompute $\mathcal{B}_{j\to i}$ without $u-v$ for each neighbor $j$
```

A. Proof of Loop-Free Convergence with FIFR++

Now, we sketch a proof that FIFR++ is loop-free, based on the following assumptions: i) There is only one failed link in the network that is protected with FIFR; ii) Only one link state update is being propagating throughout the network; iii) Each progressive update increments the failed link cost by 1;[2] iv) Links are bidirectional with symmetric costs.

Let $u-v$, with original cost of C_{u-v}, be the failed link. Its cost is progressively updated and currently it is being changed from $\overleftarrow{C}_{u-v}$ (cost in the *bc* era) to $\overrightarrow{C}_{u-v}$ (cost in the *ac* era). For other symbols too, we use the overhead left arrow to refer to the *bc* era state and right arrow to refer to that in the *ac* era.

Suppose a packet is being forwarded from source s to destination d. Without loss of generality, let us assume that a packet from s to d, if it were to cross the link $u-v$, will pass in the direction $u\to v$. In the following, we show that, under FIFR++, this packet does not get caught in a loop, i.e., does not traverse the same link in the same direction more than once.

Property 1: A packet does not loop if $u\to v \notin \mathcal{P}(\overleftarrow{\mathcal{T}}_s, d)$

If $u\to v \notin \mathcal{P}(\overleftarrow{\mathcal{T}}_s, d)$, then $u\to v \notin \mathcal{P}(\overrightarrow{\mathcal{T}}_s, d)$, i.e., if the shortest path from s to d in the *bc* topology does not include $u\to v$, then same is the case in the *ac* topology too. This is true because the path through $u\to v$ gets costlier as the cost C_{u-v} is increased from *bc* to *ac*. Since this is the only change between the *bc* and *ac* topologies, $\mathcal{P}(\overleftarrow{\mathcal{T}}_s, d) = \mathcal{P}(\overrightarrow{\mathcal{T}}_s, d)$. Therefore, all routers along that path forward the packet consistently to d.

Property 2: A packet does not loop if it is not rerouted by u.

As per FIFR++, a packet may be forwarded using back hops in $\mathcal{B}$ tables, only after it reaches u and gets rerouted. In all other cases, it is forwarded using usual next hops in $\mathcal{F}$ tables. Suppose that is not the case and a router i forwards the packet using the back hop entry $\mathcal{B}_{j\to i}^d$. For this to happen, i should be a next hop to d according to j and vice versa. Obviously this can not happen if i and j are in the same era.

Let us assume that router i is in the *ac* era and j in the *bc* era. Then, in j's view of the topology, $\mathcal{D}(\overleftarrow{\mathcal{T}}_j, d) > \mathcal{D}(\overleftarrow{\mathcal{T}}_i, d)$ and in i's view, $\mathcal{D}(\overrightarrow{\mathcal{T}}_i, d) > \mathcal{D}(\overrightarrow{\mathcal{T}}_j, d)$. This is not possible considering that $\mathcal{D}(\overrightarrow{\mathcal{T}}_j, d) = \mathcal{D}(\overleftarrow{\mathcal{T}}_j, d)$ or $\mathcal{D}(\overrightarrow{\mathcal{T}}_j, d) = \mathcal{D}(\overleftarrow{\mathcal{T}}_j, d)+1$, and $\mathcal{D}(\overrightarrow{\mathcal{T}}_i, d) = \mathcal{D}(\overleftarrow{\mathcal{T}}_i, d)$ or $\mathcal{D}(\overrightarrow{\mathcal{T}}_i, d) = \mathcal{D}(\overleftarrow{\mathcal{T}}_i, d) + 1$. Similarly, when i is in the *bc* era and j in the *ac* era, no back hops are used to forward the packet.

In other words, when a packet is not rerouted by u, all routers along the path forward it using $\mathcal{F}$ tables only and FIFR plays no role in forwarding it. This scenario is already shown be loop-free due to progressive link metric increments [8].

Property 3: A packet does not loop if it is rerouted by u.

The path taken by a packet rerouted by u can be split into 3 segments: a forward segment to the point of failure $s\rightsquigarrow u$, a backward segment $u\rightsquigarrow x\to y$ to a *turning point* y (where it switches from backwarding to forwarding), and an additional forward segment $y\to z\rightsquigarrow d$ to destination. Note that s and x

[1]For convenience, we refer to the cost of a link that is down as ∞. In practice, however, it is set to the maximum possible metric value.

[2]Using the arguments analogus to that in [8], which shows that updating with optimized metric sequence is loop-free, it is possible to prove that similar sequence with larger metric increments also does not cause loops with FIFR.

can be u itself and similarly z can be same as d. It follows from Property 2 that no looping occurs in the path $s \rightsquigarrow u$. Next, we prove that the other two segments are also loop-free.

From the properties of FIFR, it is known that both the segments are loop-free in the original topology (with original cost of C_{u-v} for $u-v$). Since back hops $\mathcal{B}$ are based on the original topology, backwarding is done consistently by the routers following u. The only deviation is that some unusual incoming interfaces with associated back hops in the original topology may no longer be unusual incoming interfaces in the *bc* or *ac* topology. Without loss of generality, let y be the turning point from backwarding segment to forwarding segment.

We show that y can *safely* switch from using backhops, $\mathcal{B}^d_{x \to y}$, to either $\overleftarrow{\mathcal{F}}^d_y$ or $\overrightarrow{\mathcal{F}}^d_y$, when x is no longer a next hop from y to d. First, consider the scenario when router y is in the *bc* era. Let z be the next hop from y to d in the *bc* topology (with cost $\overleftarrow{C}_{u-v}$ greater than original cost of C_{u-v} for link $u-v$). Then, we can show that the packet from z to d would not arrive back at u, avoiding any potential for looping. For z to be a next hop from y in the *bc* topology, we must have $\mathcal{D}(\overleftarrow{\mathcal{T}}_x, d) + C_{x-y} > \mathcal{D}(\overleftarrow{\mathcal{T}}_z, d) + C_{y-z}$, whereas $\mathcal{D}(\mathcal{T}_x, d) + C_{x-y} < \mathcal{D}(\mathcal{T}_z, d) + C_{y-z}$ in the original topology. Considering that there is no change in the cost of any other link except $u-v$, this is possible only if $u \to v \notin \mathcal{P}(\overleftarrow{\mathcal{T}}_z, d)$, i.e., the shortest path from z to d does not pass through $u \to v$. Therefore, a packet gets forwarded from z to d without getting caught in a loop according to Property 1.

Now, suppose router y is in the *ac* era with z as its next hop to d, whereas x was the next hop in the *bc* topology. Since z or any other downstream routers may be in the *bc* era, the question is whether the packet from z to d reaches u and then gets rerouted to y, resulting in a forwarding loop. For that to happen, the backward path $u \rightsquigarrow x \to y$ must be shorter than $u \rightsquigarrow z \to y$, while the forward path $y \to z \rightsquigarrow d$ must be shorter than $y \to x \rightsquigarrow d$. This is an impossibility if $z \rightsquigarrow d$ were to pass through u, considering that link costs are symmetric.

Finally, if the failure of $u-v$ partitions the network and there is no path from s to d, then the packet is dropped, instead of getting rerouted, by u or v. Thus, FIFR++ can guarantee loop-free forwarding to reachable destinations during convergence.

B. Validation of FIFR++

We have validated the FIFR++ approach using Rocketfuel topologies [9], details of which are listed in Table V. We also used 10 random topologies generated using BRITE [10], varying the number of nodes from 25 to 125, each with average degrees of 4 and 6. We developed a customized simulator that fails one link at a time in the given topology and verifies if a packet from each *affected* (whose shortest path passes through that link) pair of source and destination nodes gets caught in a loop, considering all possible combinations where each node can be in either *bc* or *ac* state. As expected, under FIFR++, with progressive updates, no packet incurred any loop. Interestingly, even when FIFR is coupled with traditional updates, only in one of the random topologies, with 50 nodes and average degree 4, we observed forwarding loops between 4 pairs of nodes for one particular link failure. This indicates that looping scenario illustrated using Fig. 2 in Section III is uncommon. Therefore, we believe it is possible to shorten the link metric sequence with FIFR++ to guarantee loop-free forwarding while also accelerating convergence, exploring which is part of our immediate future work.

TABLE V
SUMMARY OF ROCKETFUEL TOPOLOGIES

AS Number	Name	Nodes	Edges
1221	Telstra	108	306
1239	Sprint	315	1944
1755	Ebone	87	322
3257	Tiscali	161	656
3967	Exodus	79	294
6461	Abovenet	141	748

IV. LIMITATIONS AND DISCUSSION

Link metric changes besides failures: The focus of this paper has been about dealing with failed links. But a similar approach can be used to increase the link cost to a certain target metric, instead of ∞. An issue that arises then is when to update the backwarding entries to reflect the new cost. We suggest updating backwarding table after a certain timeout interval since a previous link cost increase update. Again, this does not need any coordination between routers. Similarly, this approach can be used to introduce a new link or bring up a link that was previously down. We find that in case of decreasing link cost, both forwarding and backwarding table entries can be updated simultaneously without causing any loops.

Node failures and asymmetric link costs: Our discussion and the proof assumed that there is only one link failure in the network. The question is can FIFR++ deal with failure of multiple links attached to a single router. Since both FIFR and progressive updates can individually handle router failures, we plan to investigate that. Similarly, this paper assumes links are bidirectional with symmetric link costs. We plan to expand on this work to deal with asymmetric link weights too.

Optimization of link metric sequence: As mentioned earlier, back hops under FIFR, meant for facilitating fast reroute, can help mitigate potential forwarding loops during convergence. However, they can not completely prevent them, necessitating progressive link updates. This work used the metric sequence output by the method in [8], without taking into account the natural capabilities of FIFR in mitigating loops. We see significant potential in shortening the metric sequence with the knowledge of FIFR and we plan to exploit this potential.

V. Related Work

An approach most related to this work uses the combination of NotVia [1] and interface specific forwarding [11] for loop-free convergence and fast reroute. The relative merit of FIFR++, unlike NotVia which relies on encapsulation, is that it provides the same service without any modification to the IP datagram.

Safeguard [12] aims to provide both fast route and transient loop prevention, by carrying remaining path length in all packets. Compared with such schemes, FIFR++ provides failure resilience without additional information in the packet header.

Along the lines of progressive updates, [13] and [14] provide efficient algorithms for determining metric sequences for shutting down a link or a router. Similar to [8], they are, however, meant for planned maintenance. FIFR++ can leverage them and handle both unplanned and planned failures gracefully.

An approach for mitigating transient forwarding loops during convergence using interface-specific forwarding was proposed in [15]. But this method prevents loops by discarding packets that arrive through unusual interfaces. In contrast, FIFR++ ensures loop-free convergence without any packet loss.

Failure carrying packets [16] does away with the need for convergence by carrying the list of failed links in the packet. Packet Recycling [17] can deal with more than one failure by rerouting packets leveraging cellular graph embeddings, but needs multiple bits in the header. Both these require a significant modification to the packet header.

Software defined networking (SDN), with a centralized control plane, has been gaining in popularity [18]. Yet, there have been attempts to combine the advantages of SDN and traditional routing. In [19], Vissicchio et al propose a hybrid approach called Fibbing, which introduces fake nodes in the network to induce the desired forwarding tables. Since link failures will necessitate changes in the forwarding plane and fake nodes, we believe approaches like FIFR++ for providing failure protection and loop-free convergence are still relevant.

A recent work [20] shows resiliency to 4 simultaneous failures using interface-specific forwarding. We plan to study this approach and extend FIFR++ to provide loop-free forwarding and convergence even in the presence of multiple failures.

VI. Conclusion

This paper investigated the possibility of using failure inference based fast reroute (FIFR) coupled with a link state update mechanism for loop-free convergence while performing fast reroute, without making any modification to the IP datagram or requiring coordination between routers. We found that FIFR mitigates potential loops during convergence with traditional link state updates. However, progressive link metric updates need to be employed in conjunction with FIFR to guarantee loop-free convergence. We proved that this combined approach of FIFR++ is loop-free and validated it using Rocketfuel and random topologies. Based on encouraging results about FIFR even without progressive updates, we are currently devising ways to determine the optimized metric sequence for FIFR++ that can lead to faster loop-free convergence.

References

[1] S. Bryant, S. Previdi, and M. Shand, “A framework for IP and MPLS fast reroute using not-via addresses,” RFC 6981, Aug. 2013.

[2] A. Kvalbein, et al, “Fast IP network recovery using multiple routing configurations,” in *IEEE Infocom*, Apr. 2006.

[3] J. Wang and S. Nelakuditi, “IP Fast Reroute with Failure Inferencing,” in *INM*, Aug. 2007.

[4] G. Retvari, J. Tapolcai, G. Enyedi, and A. Csaszar, “Ip fast reroute: Loop free alternates revisited,” in *INFOCOM*, April 2011, pp. 2948–2956.

[5] S. Bryant, C. Filsfils, S. Previdi, M. Shand, and N. So, “Remote Loop-Free Alternate (LFA) Fast Reroute (FRR),” RFC 7490, Apr. 2015.

[6] S. Nelakuditi, S. Lee, Y. Yu, Z.-L. Zhang, and C.-N. Chuah, “Fast Local Rerouting for Handling Transient Link Failures,” *IEEE/ACM Trans. Networking*, vol. 15, no. 2, pp. 359–372, Apr. 2007.

[7] P. Francois and O. Bonaventure, “Avoiding Transient Loops during IGP Convergence in IP Networks,” *ACM Transactions on Networking*, vol. 15, no. 6, pp. 1280–1292, Dec. 2007.

[8] P. Francois, M. Shand, and O. Bonaventure, “Disruption free topology reconfiguration in OSPF networks,” in *IEEE INFOCOM*, May 2007.

[9] N.Spring, R. Mahajan, and D. Wetherall, “Measureing ISP topologies with Rocketfuel,” in *ACM SIGCOMM*, Aug. 2002.

[10] A. Medina, A. Lakhina, I. Matta, and J. Byers, “Brite: An approach to universal topology generation,” in *Proceedings of MASCOTS 2001*, August 2001.

[11] G. Robertson, N. Roy, P. K. Penumarthi, S. Nelakuditi, and J. M. O’Kane, “Loop-free convergence with unordered updates,” *IEEE Transactions on Network and Service Management*, 2017.

[12] A. Li, X. Yang, and D. Wetherall, “SafeGuard: Safe Forwarding during Routing Changes,” in *ACM CoNEXT*, 2009.

[13] F. Clad, P. Merindol, J.-J. Pansiot, P. Francois, and O. Bonaventure, “Graceful convergence in link-state ip networks: A lightweight algorithm ensuring minimal operational impact,” *IEEE/ACM Trans. Netw.*, vol. 22, no. 1, pp. 300–312, Feb. 2014.

[14] F. Clad, S. Vissicchio, P. Mérindol, P. Francois, and J.-J. Pansiot, “Computing minimal update sequences for graceful router-wide reconfigurations,” *IEEE/ACM Trans. Netw.*, vol. 23, no. 5, pp. 1373–1386, Oct. 2015.

[15] S. Nelakuditi, Z. Zhong, J. Wang, R. Keralapura, and C.-N. Chuah, “Mitigating transient loops through interface-specific forwarding,” *Computer Networks*, vol. 52, no. 3, pp. 593–609, 2008.

[16] K. Lakshminarayanan, M. Caesar, M. Rangan, T. Anderson, S. Shenker, and I. Stoica, “Achieving convergence-free routing using failure-carrying packets.” in *ACM SIGCOMM*, Aug. 2007.

[17] S. S. Lor, R. Landa, and M. Rio, “Packet re-cycling: Eliminating packet losses due to network failures,” in *ACM HotNets*, Oct. 2010.

[18] N. Feamster, J. Rexford, and E. Zegura, “The road to SDN,” *Queue*, vol. 11, no. 12, p. 20, 2013.

[19] S. Vissicchio, O. Tilmans, L. Vanbever, and J. Rexford, “Central control over distributed routing,” in *ACM SIGCOMM*, Aug. 2015.

[20] M. Chiesa and I. Nikolaevskiy and S. Mitrovi and A. Gurtov and A. Madry and M. Schapira and S. Shenker, “On the resiliency of static forwarding tables,” *IEEE/ACM Transactions on Networking*, vol. 25, no. 2, pp. 1133–1146, April 2017.